Introduction to Computer Programming

Basics of Java programming

Basic form of a Java program

- Every Java program is made up of 1 or more Java classes.
- Syntax:

```
public class ClassName
{


}
```

Methods

- Each class has one or more methods.
- A **method** is a collection of instructions that are executed in the order given.
- Each Java program must have (at the very least) a `main` method.
- The `main` method is the first method executed in the program.
- The `main` method can **call** other methods to do a part of the work of the program.

Example

```
public class SimpleExample
{
  public static void main(String[] args)
  {
   System.out.print("My favorite food is ");
   System.out.println("pizza.");
  }
}
```

- `print` and `println` are methods in the `System.out` class.
- `System.out` is already written for you.

Example

```
public class SimpleExample
{
  public static void main(String[] args)
  {
   System.out.println("My favorite food is "
      + "pizza.");
  }
}
```

+ represents concatenation.

Variables

- Each method can define one or more variables to store information.
- A **variable** requires a data type and a name.
- Examples:
  - `int exitNumber;`
  - `double areaInAcres;`
  - `char parkingLot;`
  - `boolean lightSwitchOn;`
  - `String teamName;`

Variables

- Variables can be declared and initialized, or they can declared first and initialized later.
- Examples:

```
int exitNumber = 3;
double areaInAcres = 124.54;
char parkingLot;
boolean lightSwitchOn;
String teamName = "Steelers";
  ...
parkingLot = 'C';
lightSwitchOn = true;
```

Primitive Variables

- Primitive variable types are:
  - **`byte, short, int, long, float, double, char, boolean`**
- Primitive variables only hold data. They don't have any other special abilities.
- Numerical variables can only hold data values in specific ranges given by their type.
- Character variables only hold one symbol (letter, digit, punctuation, space, etc.)
- Boolean variables only hold **`true`** or **`false`** (more about **`boolean`** soon)

Numerical Primitives

TYPE	NUMBER OF BITS	RANGE OF VALUES
`byte`	8	-2^{7} to $2^{7}-1$
`Short`	16	-2^{15} to $2^{15}-1$
`int`	32	-2^{31} to $2^{31}-1$
`long`	64	-2^{63} to $2^{63}-1$
`Float`	32	~7 decimal digits
`double`	64	~15 decimal digits

Using variables

- Assignment Statement
  - *variable = expression* **;**
  - The single variable on the left side of the = sign is assigned the value of the expression on the right side of the = sign.
  - The expression can be a simple variable or a more complex calculation.
  - Generally, the data type of the variable and the expression should be the same.
    - There are a few exceptions we'll see.

Using variables

```
int x = 15;
int y = 100;
```

Use original value of x for each example.

x =

1 **`x = y;`** _________

2 **`x = x * y + 1;`** _________

3 **`x = x * (y + 1);`** _________

Using variables

```
int x = 15;
int y = 100;
```

Use original value of x for each example.

x =

4 **`x = x * y;`** _________
5 **`x *= y;`**

6 **`x = x + 1;`** _________
7 **`x += 1;`**
8 **`x++;`**

Using expressions

- We can build numerical expressions using the following arithmetic operators:
  - + addition
  - - subtraction
  - * multiplication
  - / division (more on this shortly)
  - % modulo (more on this shortly)
- Operations are evaluated based on precedence, just like regular mathematics.
  - Use parentheses to force other precedence.

Division and Modulo

```
int a = 100;          double z = 100.0;
int b = 40;           double y = 40.0;
int c;                double x;

1 c = a / b;                    _____________

2 x = z / y;                    _____________

3 c = b / a;                    _____________

4 x = y / z;                    _____________
```

Division and Modulo (cont'd)

```
int a = 100;          double z = 100.0;
int b = 40;           double y = 40.0;
int c;                double x;

5 c = a % b;                    _____________

6 c = b % a;                    _____________
```

Mixing Types

```
int a = 100;          double z = 100.0;
int b = 40;           double y = 40.0;
int c;                double x;

7 x = a / b;   ← widening       _____________

8 c = z / y;   ← narrowing      _____________

9   c = (int)(z / y);           _____________
        ↑ typecasting
```

Example

```
public class EggCalculator {
  public static void main(String[] args) {
   int numEggs = 15100;

   int numDozens = ___________________;
   System.out.print("You have " + numDozens);
   System.out.print(" dozen eggs and ");

     System.out.println(_______________________
      + " left over.");
  }
}
```

(There are 12 eggs in 1 dozen eggs.)

Round-Off Errors

- Floating point numbers are stored with a limited number of bits in computer memory.
  - Some floating point numbers may not be able to be stored exactly.
  - Example: 0.1_{10} (one-tenth)
  - In binary, one-tenth is 0.00011...
  - Start a double variable x with a value of 0.0.
  - Add 0.1 to x ten times and output x.
  - You don't get 1.0!